bidang
REKAYASA

STUDI DAN IMPLEMENTASI ALGORITMA RSA UNTUK PENGAMANAN DATA TRANSKRIP AKADEMIK MAHASISWA

TRI RAHAJOENINGROEM
MUHAMMAD ARIA

Jurusan Teknik Elektro
Universitas Komputer Indonesia

RSA adalah salah satu teknik kriptografi dimana kunci untuk melakukan enkripsi berbeda dengan kunci untuk melakukan dekripsi. Data transkrip akademik mahasiswa merupakan salah satu data yang harus dijaga keamanannya sehingga perlu diterapkan suatu teknik pengamanan dalam penyimpanannya. Pada makalah ini akan dibahas proses enkripsi (penyandian data) nilai transkrip akademik mahasiswa menggunakan algoritma RSA, lalu akan dibahas proses melakukan dekripsi (pengembalian data asli), serta akan dibahas pula proses pembangkitan kunci pada algoritma RSA ini. Kinerja yang diukur dari algoritma RSA ini waktu komputasi serta kompleksitas memori yang dibutuhkan dalam melakukan enkripsi dan dekripsi data. Sebuah perangkat lunak berbasis LabVIEW dibangun untuk implementasi algoritma RSA ini. Hasil pengujian menunjukkan bahwa algoritma RSA berhasil diimplementasikan untuk pengamanan data transkrip akademik mahasiswa. Berdasarkan pengujian diperoleh waktu komputasi algoritma RSA adalah sebesar 15625 mikrodetik. Sedangkan kompleksitas memori yang dibutuhkan algoritma RSA sebesar 3908 bytes.

Kata kunci *–Algoritma RSA, Algoritma Euclid, Cipher Block Chaining, Enkripsi, Dekripsi*

PENDAHULUAN

Masalah keamanan dan kerahasiaan data merupakan hal yang penting dalam suatu organisasi. Data yang bersifat rahasia tersebut perlu dibuatkan sistem penyimpanan dan pengirimannya agar tidak terbaca atau diubah oleh orang-orang yang tidak bertanggung jawab, baik saat data tersebut tersimpan sebagai *file* di dalam komputer maupun saat data tersebut dikirim melalui email. Untuk menyimpan data tersebut agar benar-benar aman, tentunya dilakukan sistem pengamanan yang baik, yang bebas dari jangkauan orang-orang yang tidak berhak, baik bebas dari jangkauan secara fisik maupun secara sistem. Untuk bebas secara fisik, maka faktor orang sebagai penjaga memegang peranan yang penting, sedangkan aman menurut sistem adalah dokumen tersebut tersimpan dalam kondisi yang tidak dapat dibaca oleh orang yang tidak berhak. Apalagi jika data tersebut berada dalam suatu jaringan komputer yang terhubung/terkoneksi dengan jaringan internet. Tentu saja data yang penting tersebut tidak boleh diketahui apalagi diubah oleh pihak yang tidak berwenang.

Untuk membangun sistem penyimpanan data transkrip nilai mahasiswa yang hasil simpanannya tidak dapat dibaca oleh orang, dalam penelitian ini telah dikembangkan model sistem pengamanan dengan proses enkripsi dan dekripsi menggunakan algo-

ritma RSA. Tujuan-tujuan dari penggunaan algoritma kriptografi antara lain adalah sebagai berikut.

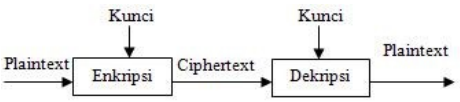
- *Confidentiality*, yaitu menjaga kerahasiaan pesan dan menyimpan data dengan menyandikan informasi menggunakan teknik-teknik enkripsi
- *Message Integrity*, yaitu memberikan jaminan bahwa untuk tiap bagian pesan tidak akan mengalami perubahan dari saat data dibuat/dikirim oleh pengirim sampai dengan saat data tersebut dibuka oleh penerima data
- *Non-repudiation*, yaitu memberikan cara untuk membuktikan bahwa suatu dokumen datang dari seseorang tertentu sehingga apabila ada seseorang yang mencoba mengakui memiliki dokumen tersebut, dapat dibuktikan kebenarannya dari pengakuan orang tersebut
- *Authentication*, yaitu memberikan dua layanan. Pertama mengidentifikasikan keaslian suatu pesan dan memberikan jaminan keotentikannya. Kedua untuk menguji identitas seseorang apabila ia akan memasuki sebuah sistem.

Susunan makalah ini yaitu pada bagian 2, dijelaskan mengenai teori dasar algoritma kriptografi RSA, teori matematika yang mendukung beserta mode operasi CBC. Pada bagian 3, dibahas arsitektur RSA yang dirancang agar dapat diaplikasikan untuk mengenkripsi data transkrip akademik mahasiswa. Bagian 4 menyajikan hasil implementasi dan pengujian data, dan bagian 5 adalah kesimpulan

TEORI DASAR

Algoritma RSA

Kriptografi bertujuan menjaga kerahasiaan informasi yang terkandung dalam data sehingga informasi tersebut tidak dapat diketahui oleh pihak yang tidak sah. Dalam menjaga kerahasiaan data, kriptografi mentransformasikan data jelas (*plaintext*) ke dalam bentuk data sandi (*ciphertext*) yang tidak dapat dikenali. *Ciphertext* inilah yang kemudian dikirimkan oleh pengirim (*sender*) kepada penerima (*receiver*). Setelah sampai di penerima, *ciphertext* tersebut ditransformasikan kembali ke dalam bentuk *plaintext* agar dapat dikenali. Proses tranformasi dari *plaintext* menjadi *ciphertext* disebut proses *Encipherment* atau enkripsi (*encryption*), sedangkan proses mentransformasikan kembali *ciphertext* menjadi *plaintext* disebut proses dekripsi (*decryption*). Kriptografi menggunakan suatu algoritma (*cipher*) dan kunci (*key*). *Cipher* adalah fungsi matematika yang digunakan untuk mengenkripsi dan mendekripsi. Sedangkan kunci merupakan sederetan bit yang diperlukan untuk mengenkripsi dan mendekripsi data. Secara sederhana istilah-istilah di atas dapat digambarkan sebagai berikut.



Gambar 1.
Proses Enkripsi/Dekripsi Sederhana

Berdasarkan kunci yang digunakan, algoritma kriptografi dapat dibedakan atas dua golongan, yaitu :

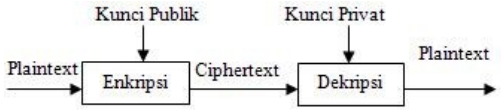
a. Symmetric Algorithms

Algoritma kriptografi simetris atau disebut juga algoritma kriptografi konvensional adalah algoritma yang menggunakan kunci untuk proses enkripsi sama dengan kunci untuk proses dekripsi.

b. Asymmetric Algorithms

Algoritma kriptografi nirsimetris adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsinya. Algoritma ini disebut juga algoritma kunci umum (*public key algorithm*) karena kunci untuk enkripsi dibuat umum (*public key*) atau dapat diketahui oleh setiap orang, tapi kunci untuk dekripsi hanya diketahui oleh orang yang berwenang mengetahui data yang disandikan atau sering disebut kunci pribadi (*private key*). Contoh algoritma terkenal yang menggunakan kunci

asimetris adalah RSA dan ECC.



Gambar 2. Proses Enkripsi/Dekripsi Kriptografi Kunci Publik

Algoritma RSA dibuat oleh tiga orang peneliti dari MIT (*Massachussets Institute of Technology*) pada tahun 1976, yaitu Ron Rivest, Adi Shamir dan Leonard Adleman. RSA adalah salah satu teknik kriptografi dimana kunci untuk melakukan enkripsi berbeda dengan kunci untuk melakukan dekripsi. Kunci untuk melakukan enkripsi disebut sebagai kunci publik, sedangkan kunci untuk melakukan dekripsi disebut sebagai kunci privat. Orang yang mempunyai kunci publik dapat melakukan enkripsi tetapi yang dalam melakukan dekripsi hanyalah orang yang memiliki kunci privat. Kunci publik dapat dimiliki oleh sembarang orang, tetapi kunci privat hanya dimiliki oleh orang tertentu saja.

Untuk pembangkitan pasangan kunci RSA, digunakan algoritma sebagai berikut.

1. Dipilih dua buah bilangan prima sembarang yang besar, *p* dan *q*. Nilai *p* dan *q* harus dirahasiakan.
2. Dihitung $n = p \times q$. Besaran *n* tidak perlu dirahasiakan.
3. Dihitung $m = (p - 1)(q - 1)$
4. Dipilih sebuah bilangan bulat sebagai kunci publik, disebut namanya *e*, yang relatif prima terhadap *m*.
   e relatif prima terhadap *m* artinya faktor pembagi terbesar keduanya adalah 1, secara matematis disebut gcd (e,m) = 1. Untuk mencarinya dapat digunakan algoritma Euclid
5. Dihitung kunci privat, disebut namanya *d* sedemikian agar $(d \times e) \bmod m = 1$.
   Untuk mencari nilai *d* yang sesuai dapat juga digunakan algoritma Extended Euclid.

Maka hasil dari algoritma tersebut diperoleh :

- kunci publik adalah pasangan *(e,n)*
- kunci privat adalah pasangan *(e,m)*

n tidak bersifat rahasia, namun ia diperlukan pada perhitungan enkripsi/dekripsi.

Keamanan algoritma RSA terletak pada tingkat kesulitan dalam memfaktorkan bilangan non prima menjadi faktor primanya, yang dalam hal ini $n = p \times q$. Jika *n* berhasil difaktorkan menjadi *p* dan *q*, maka $m = (p - 1)(q - 1)$ dapat dihitung. Dan karena kunci enkripsi *e* telah diumumkan (tidak dirahasiakan), maka kunci dekripsi *d* dapat dihitung melalui persamaan $(d \times e) \bmod n = 1$. Selama belum ditemukan cara untuk memfaktorkan bilangan besar menjadi faktor-faktor primanya, maka selama itu pula keamanan algoritma RSA terjamin.

Penemu algoritma RSA menyarankan nilai *p* dan *q* panjangnya lebih dari 100 digit. Dengan demikian hasil kali $n = p \times q$ akan berukuran lebih dari 200 digit. Dengan asumsi bahwa algoritma pemfaktoran yang digunakan adalah algoritma yang tercepat saat ini dan komputer yang dipakai mempunyai kecepatan 1 milidetik, menurut Rivest dan kawan-kawan, usaha untuk mencari faktor bilangan 200 digit membutuhkan waktu komputasi selama 4 milyar tahun.

Untuk algoritma enkripsi menggunakan RSA adalah sebagai berikut.

Plaintext M dinyatakan menjadi blok-blok $m_1, m_2, \ldots$ sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang $[0, n - 1]$

(harus dipenuhi persyaratan bahwa nilai m_i harus terletak dalam himpunan nilai 0, 1, 2, ..., $n - 1$ untuk menjamin hasil perhitungan tidak berada di luar himpunan)

Setiap blok m_i dienkripsikan menjadi blok c_i dengan rumus $c_i = m_i^e \bmod n$

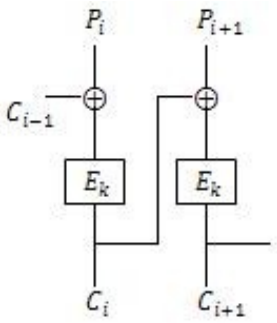
Sedangkan untuk dekripsi, maka setiap blok *ciphertext* c_i didekripsikan kembali menjadi blok m_i dengan rumus $m_i = c_i^d \bmod n$.

Mode Operasi Chiper Block Chaining

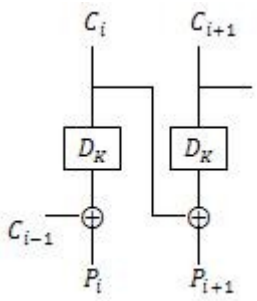
Karena akan diimplementasikan untuk menyandikan data transkrip akademik mahasiswa, algoritma RSA ini akan dioperasikan dalam mode *Chiper Block Chaining* (CBC). CBC adalah mode yang

dapat digunakan untuk melakukan enkripsi *cipher* blok. Pada *cipher* blok, rangkaian bit-bit *plaintext* dibagi menjadi blok-blok bit dengan panjang yang sama. Enkripsi dilakukan terhadap blok bit *plaintext* menggunakan bit-bit kunci (yang ukurannya sama dengan blok *plaintext*). Algoritma enkripsi menghasilkan blok *ciphertext* yang berukuran sama dengan blok *plaintext*. Dekripsi dilakukan dengan cara yang serupa seperti enkripsi. Operasi *Chiper Block Chaining* merupakan penerapan mekanisme umpan balik pada sebuah blok bit dimana hasil enkripsi blok sebelumnya diumpanbalikkan ke dalam proses enkripsi blok saat ini (*current*). Caranya, blok *plaintext* yang *current* di-XOR-kan terlebih dahulu dengan blok *ciphertext* hasil enkripsi sebelumnya, selanjutnya hasil peng-XOR-an ini masuk ke dalam fungsi enkripsi.

Dengan mode CBC, setiap blok *ciphertext* bergantung tidak hanya pada blok *plaintext*nya tetapi juga pada seluruh blok *plaintext* sebelumnya. Dekripsi dilakukan dengan memasukkan blok *ciphertext* yang *current* ke fungsi dekripsi, kemudian meng-XOR-kan hasilnya dengan blok *ciphertext* sebelumnya. Dalam hal ini, blok *ciphertext* sebelumnya berfungsi sebagai umpan-maju pada akhir proses dekripsi. Ilustrasi dari proses ini enkripsi ditujukkan melalui Gambar 3. Sedangkan ilustrasi dari proses dekripsi diilustrasikan melalui Gambar 4.



Gambar 3.
Ilustrasi Enkripsi dengan Mode CBC



Gambar 4.
Ilustrasi Dekripsi dengan Mode CBC

Blok diagramE_k menunjukkan proses enkripsi, sedangkan blok diagram D_k menunjukkan proses dekripsi.

Secara matematis, enkripsi dengan mode CBC dinyatakan sebagai

$$C_i = E_k (XOR (P_i , C_{i-1})$$

Dan dekripsi sebagai

$$P_i = D_k (XOR (C_i, C_{i-1})$$

Dalam hal ini $C_0 = IV$ (*initialization vector)* dapat diberikan oleh pengguna atau dibangkitkan secara acak oleh program. *IV* tidak mempunyai makna, ia hanya digunakan untuk membuat tiap blok *ciphertext* menjadi unik.

Contoh RSA

Akan dilakukan enkripsi menggunakan algoritma RSA terhadap *plaintext M = FISIKA*. Pertama-tama *plaintext* tersebut diubah menjadi format ASCII sebagai berikut :

Text (karakter)	F	I	S	I	K	A
ASCII (heksa)	46	49	53	49	4B	41
ASCII (desimal)	70	73	83	73	75	65

Plaintext dalam format ASCII desimal tersebut kemudian dipecah menjadi blok-blok tiga digit berikut :

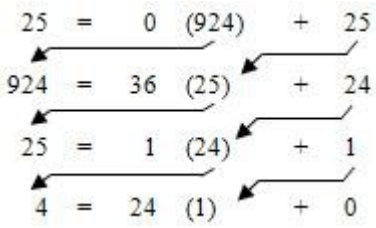
$m_i = 707$ $m_3 = 737$

$m_2 = 383$ $m_4 = 565$

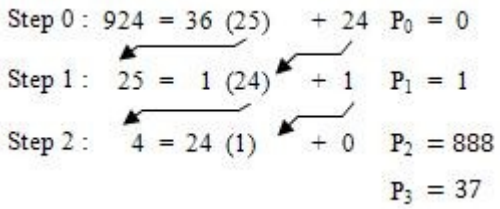
Dalam membuat kunci RSA, perlu dirancang agar nilai m_i masih terletak di dalam rentang antara 0 sampai $n - 1$. Maka ditentukan bahwa nilai n minimal adalah 909. Nilai ini diambil berdasarkan pertimbangan bahwa karakter huruf kapital dengan nilai terbesar adalah Z dengan nilai ASCII yaitu 5Ah atau 90. Kombinasi ZZ akan dapat dipecah menjadi blok 909 atau 090.
Misalkan dipilih $p = 23$ dan $q = 43$ (keduanya prima), maka dapat dihitung

$$n = p \times q = 989$$
$$m = (p - 1) \times (q - 1) = 924$$

Dipilih kunci publik $e = 25$ (yang relatif prima dengan 924 karena pembagi bersama terbesarnya adalah 1). Bahwa 25 relatif prima terhadap 924 dapat dibuktikan dengan mencari nilai gcd (25,924) melalui algoritma Euclid seperti berikut.



Hasil gcd pada algoritma ini adalah hasil sisa bagi terakhir sebelum 0. Maka pada perhitungan diatas terlihat bahwa sisa bagi sebelum nol adalah 1. Maka gcd (25, 924) = 1.
Selanjutnya untuk menghitung kunci privat d algoritma Extended Euclid sebagai berikut.



P_2 dan P_3 dihitung melalui persamaan berikut

$P_2 = (p_{i-2} - p_{i-1}q_{i-2}) \bmod n$
$= (0 - 1(36)) \bmod 924 = 888$
$P_3 = (1 - 888(1)) \bmod 924 = 37$

Maka diperoleh kunci publik adalah 25 dan 989. Sedangkan kunci privat adalah 37 dan 989. Enkripsi setiap blok diperoleh menggunakan kunci public 25 dan 989 dengan cara sebagai berikut :

$$c_1 = 707^{25} \bmod 989$$

Untuk menghitung $707^{25} \bmod 989$ dapat menggunakan teknik *divide and conquer* untuk membagi pemangkatnya sampai berukuran kecil. Ilustarinya adalah sebagai berikut.

$$707^{25} = 707^{16} \cdot 707^{8} \cdot 707^{1}$$

$$707^{2} \bmod 989 = 499849 \bmod 989 = 404$$

$$\begin{aligned} 707^{4} \bmod 989 &= (707^{2} \cdot 707^{2}) \bmod 989 \\ &= [(707^{2} \bmod 989)(707^{2} \bmod 989)] \\ &\quad \bmod 989 \\ &= (404 \cdot 404) \bmod 989 \\ &= 163216 \bmod 989 = 31 \end{aligned}$$

$$\begin{aligned} 707^{8} \bmod 989 &= (707^{4} \cdot 707^{4}) \bmod 989 \\ &= [(707^{4} \bmod 989)(707^{4} \bmod 989)] \\ &\quad \bmod 989 \\ &= (31 \cdot 31) \bmod 989 \\ &= 961 \bmod 989 = 961 \end{aligned}$$

$$\begin{aligned} 707^{16} \bmod 989 &= (707^{8} \cdot 707^{8}) \bmod 989 \\ &= [(707^{8} \bmod 989)(707^{8} \bmod 989)] \\ &\quad \bmod 989 \\ &= (961 \cdot 961) \bmod 989 \\ &= 923521 \bmod 989 = 784 \end{aligned}$$

$$\begin{aligned} 707^{25} \bmod 989 &= 707^{16} \cdot 707^{8} \cdot 707^{1} \bmod 989 \\ &= ((707^{16} \bmod 989 \cdot 707^{8} \bmod 989) \bmod 989 \\ &\quad \cdot 707^{1} \bmod 989) \bmod 989 \\ &= ((784 \cdot 961) \bmod 989 \cdot 707) \bmod 989 \\ &= ((753424) \bmod 989 \cdot 707) \bmod 989 \\ &= (795 \cdot 707) \bmod 989 = 313 \end{aligned}$$

Jadi $c_1 = 707^{25} \bmod 989 = 313$.
Dengan cara yang sama dapat diperoleh :

$c_2 = 383^{25} \bmod 989 = 776$
$c_3 = 737^{25} \bmod 989 = 737$
$c_4 = 565^{25} \bmod 989 = 909$

Maka *ciphertext* adalah C = 313 776 737 909
Perlu diingat, bahwa *ciphertext* ini dalam format ASCII desimal. Jika diubah kembali

menjadi format karakter maka dapat diperoleh:

ASCII (desimal)	31	37	76	73	79	09
ASCII (heksa)	1F	24	4C	49	4F	09
Text (karakter)		%	L	I	O	

Heksadesimal 1F dan 09 adalah *non-printing characters*.

Untuk melakukan dekripsi (mengubah *ciphertext* menjadi *plaintext*) maka digunakan kunci privat 37 dan 989 dengan cara sebagai berikut :

$m_1 = 313^{37} \bmod 989 = 707$
$m_2 = 776^{37} \bmod 989 = 383$
$m_3 = 737^{37} \bmod 989 = 737$
$m_4 = 909^{37} \bmod 989 = 565$

Maka diperoleh *plaintext* 707 383 737 565

Tabel 1.
Data Uji Transkrip Akademik Mahasiswa

Nama	:	Muhammad Ismail		
NIM	:	23206019		
Tempat/Tanggal Lahir	:	Bandung/27 September 1990		
No	**KODE**	**MATA KULIAH**	**SKS**	**NILAI**
1	TE38365	PLC	2	A
2	TE38362	Artificial Intelligence	3	B
3	TE37361	Kendali Cerdas	3	A
4	TE38363	Kendali Adaptif	3	B
5	TE38203	Ekonomi Teknik	2	B
6	TE36317	Mikroprosesor	2	C
7	TE34205	Rangkaian Listrik II	3	A
8	TE33203	Probabilitas dan Statistik	2	C

Tabel 2.
Daftar Data Mahasiswa

NIM	NAMA	Tempat/Tanggal Lahir
23206016	Aulia Rachman S	Bandung/ 8 Januari 1992
23206017	Fariz Gania Nugraha	Garut/10 April 1990
23206018	Mukhlis Silmi Kaffah	Bandung/12 April 1991
23206019	Muhammad Ismail	Bandung/27 September 1990

Tabel 3.
Daftar Mata Kuliah

KODE	MATA KULIAH	SKS
TE38365	PLC	2
TE38364	Estimasi dan Identifikasi	3
TE38363	Kendali Adaptif	3
TE38362	Artificial Intelligent	3
TE37361	Kendali Cerdas	3

ARSITEKTUR PENGAMANAN DATA TRANSKRIP AKADEMIK MAHASISWA MENGGUNAKAN ALGORITMA RSA

Pertama diambil data uji transkrip akademik mahasiswa seperti pada **Tabel 1**. Pada basis data akademik, telah terekam data daftar NIM beserta nama dan tempat tanggal lahir setiap mahasiswa seperti **Tabel 2.** serta daftar mata kuliah beserta kode dan SKS mata kuliah tersebut seperti **Tabel 3.** sehingga data yang perlu disimpan cukup NIM beserta kode mata kuliah dan nilai mata kuliah saja. Dengan bantuan tabel data, dengan mengetahui NIM dapat diketahui nama dan tempat/tanggal lahir mahasiswa bersangkutan dan dengan mengetahui kode mata kuliah dapat diketahui nama mata kuliah beserta beban SKS-nya.

Sebelum dimasukkan pada algoritma kriptografi RSA, NIM, kode mata kuliah dan nilai mata kuliah tersebut dikelompokkan menjadi sebagai berikut :

23206019
TE38365A
TE38362B
TE37361A
TE38363B
TE38203B
TE36317C
TE34205A
TE33203C

Pengelompokan tersebut dibuat agar setiap baris mengandung 8 karakter huruf. Sehingga kode mata kuliah digabungkan dengan nilai mata kuliah. Lalu setiap karakter dari data di atas diubah ke dalam format ASCII dalam bentuk **heksa** sehingga menjadi sebagai berikut

3233323036303139
5445333833363541
5445333833363242
5445333733363141
5445333833363342
5445333832303342
5445333633313743
5445333432303541
5445333332303343

Kemudian dibagi menjadi kelompok-kelompok lagi yang setiap kelompoknya terdiri dari 4 karakter sebagai berikut.

3233 3230 3630 3139
5445 3338 3336 3541
5445 3338 3336 3242
5445 3337 3336 3141
5445 3338 3336 3342
5445 3338 3230 3342
5445 3336 3331 3743
5445 3334 3230 3541
5445 3333 3230 3343

Nilai-nilai pada blok teks di atas masih dalam bentuk heksadesimal. Jika diubah menjadi desimal maka akan diperoleh nilai-nilai blok sebagai berikut.

12851 12848 13872 12601
21573 13112 13110 13633
21573 13112 13110 12866
21573 13111 13110 12609
21573 13112 13110 13122
21573 13112 12848 13122
21573 13110 13105 14147
21573 13108 12848 13633
21573 13107 12848 13123

Algoritma kriptografi RSA akan dioperasikan untuk melakukan enkripsi per setiap kelompok data yang terdiri dari 4 karakter tersebut. Maka rentang nilai terbesar yang akan disandikan oleh algoritma RSA adalah FFFFh (heksa) atau dalam nilai desimal adalah 65535. Maka akan dicari nilai modulus n untuk algoritma RSA dengan nilai minimal adalah 65535 agar dapat menyandikan nilai hingga 65535. Karena alasan komputasi maka penulis perlu memilih kunci yang memiliki nilai e dan d yang minimal tetapi memiliki nilai n minimal 65535.

Langkah-langkah yang dilakukan adalah sebagai berikut .

1. Dicari alternatif-alternatif nilai p dan q dalam rentang 200 hingga 300. Maka diperoleh alternatif nilai p dan q adalah 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283 dan 293.
   Berarti akan ada 256 (16 x 16) kemungkinan nilai n maupun nilai m.
2. Secara berurutan dipilih nilai p dan q

yang memberikan nilai n lebih dari 65535.

3. Dari nilai p dan q yang terpilih pada langkah 2, akan dicari nilai m.
4. Setelah suatu nilai m diperoleh maka akan dicari alternatif-alternatif nilai e untuk harga m tersebut. Menggunakan algoritma Extended Euclidean, untuk setiap alternatif nilai e akan dicari pasangan nilai d-nya. Maka untuk setiap nilai m, akan diperoleh sekumpulan alternatif pasangan nilai e dan d. Dari pasangan alternatif tersebut akan dipilih nilai e dan d yang memiliki nilai $e^2 + d^2$ terkecil. Maka untuk setiap nilai m akan diperoleh alternatif nilai e dan d nya.
5. Langkah 2 – 4 akan diulang hingga diperoleh kumpulan alternative p, q, m, e dan d seperti pada Tabel 4 berikut.
6. Maka dipilih e = 163, d = 403 dan n = 66203

Proses di atas penulis lakukan dengan membuat program berbasis LabVIEW 7 dengan tampilan seperti Gambar 5.

Tabel 4.
Alternatif Pemilihan Nilai p,q,m,e & d

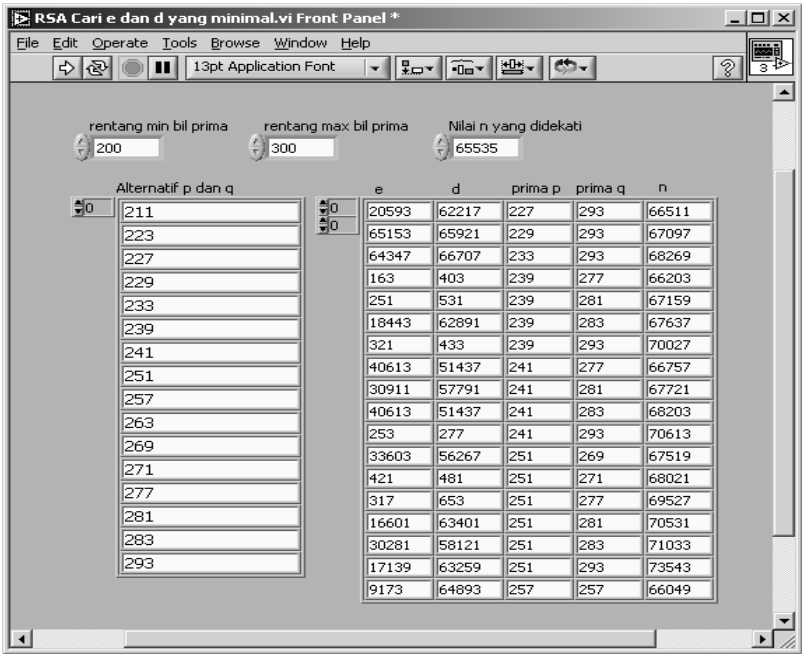
p	q	n	m	e	d
227	293	66511	65992	20593	62217
229	293	67097	66576	65153	65921
233	293	68269	67744	64347	66707
239	277	66203	65688	163	403
239	281	67159	66640	251	531
239	283	67637	67116	18443	62891
239	293	70027	69496	321	433
241	277	66757	66240	40613	51437
241	281	67721	67200	30911	57791
241	283	68203	67680	40613	51437

Blok-blok teks yang akan dienkripsikan ditunjukkan pada Tabel 5. Jika kunci publik yang telah diperoleh langsung diterapkan pada blok-blok teks tersebut, maka dapat dilakukan menggunakan ilustrasi perhitungan sebagai berikut.

$c_{11} = 12851^{163} \bmod 66203 = 6786$

Maka dapat diperoleh blok-blok teks hasil enkripsi seperti pada Tabel 6.

Dari hasil enkripsi tersebut terdapat hal



Gambar 5.
Tampilan Program LabVIEW untuk Mencari Nilai e dan d yang Optimal

Tabel 5.
Blok Teks yang akan Dienkripsikan

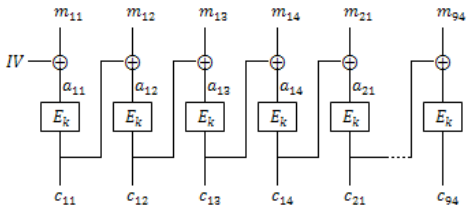
m_{11} = 12851	m_{12} = 12848	m_{13} = 13872	m_{14} = 12601
m_{21} = 21573	m_{22} = 13112	m_{23} = 13110	m_{24} = 13633
m_{31} = 21573	m_{32} = 13112	m_{33} = 13110	m_{34} = 12866
m_{41} = 21573	m_{42} = 13111	m_{43} = 13110	m_{44} = 12609
m_{51} = 21573	m_{52} = 13112	m_{53} = 13110	m_{54} = 13112
m_{61} = 21573	m_{62} = 13112	m_{63} = 12848	m_{64} = 13112
m_{71} = 21573	m_{72} = 13110	m_{73} = 13105	m_{74} = 14147
m_{81} = 21573	m_{82} = 13108	m_{83} = 12848	m_{84} = 13633
m_{91} = 21573	m_{92} = 13107	m_{93} = 12848	m_{94} = 13123

Tabel 6.
Blok Teks Hasil Enkripsi

c_{11} = 6786	c_{12} = 44229	c_{13} = 40252	c_{14} = 9312
c_{21} = 11389	c_{22} = 24330	c_{23} = 63670	c_{24} = 35950
c_{31} = 11389	c_{32} = 24330	c_{33} = 63670	c_{34} = 37217
c_{41} = 11389	c_{42} = 6235	c_{43} = 63670	c_{44} = 32040
c_{51} = 11389	c_{52} = 24330	c_{53} = 63670	c_{54} = 51934
c_{61} = 11389	c_{62} = 24330	c_{63} = 44229	c_{64} = 51934
c_{71} = 11389	c_{72} = 63670	c_{73} = 66136	c_{74} = 54165
c_{81} = 11389	c_{82} = 55592	c_{83} = 44229	c_{84} = 35950
c_{91} = 11389	c_{92} = 57609	c_{93} = 44229	c_{94} = 11583

yang tidak diinginkan. Yaitu terdapat perulangan nilai seperti pada c_{21} – c_{91}. Hal ini dapat menjadi celah bagi para kriptanalis untuk mengeksploitasi susunan bahasa yang digunakan.

Untuk mengatasi masalah ini maka akan digunakan operasi *Chiper Block Chaining* pada algoritma kriptografi RSA yang diterapkan. Rancangan arsitektur operasi CBC ditunjukkan melalui Gambar 6.



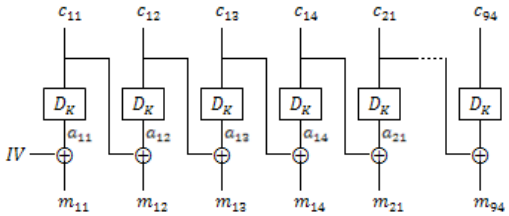
Gambar 6.
Arsitektur CBC yang Dirancang
untuk Proses Enkripsi

Dimana a_{11} adalah hasil operasi XOR antara *IV* dan m_{11} , a_{12} adalah hasil operasi XOR antara c_{11} dan m_{12}; dan seterusnya. Nilai *IV* diambil dari 2 angka terakhir NIM. Pada ilustrasi kasus ini, NIM yang digunakan adalah 23206019. Maka nilai *IV*-nya adalah 19 yang jika dikonversikan ke dalam heksadesimal adalah 3139. Hasil proses enkripsi algoritma RSA menggunakan operasi CBC ditunjukkan melalui Tabel 7.

Adapun untuk melakukan dekripsi, rancangan arsitektur CBC yang digunakan ditunjukkan melalui Gambar 7.

Tabel 7.
Proses Enkripsi Algoritma RSA Menggunakan Operasi CBC

Indeks	Format desimal			Format heksa		
	m	a	c	m	a	c
11	12851	778	48772	3233	030A	BE84
12	12848	36020	44009	3230	8CB4	ABE9
13	13872	40409	53493	3630	9DD9	D0F5
14	12601	57804	51884	3139	E1CC	CAAC
21	21573	40681	11972	5445	9EE9	2EC4
22	13112	7676	65002	3338	1DFC	FDEA
23	13110	52956	5224	3336	CEDC	1468
24	13633	8489	47586	3541	2129	B9E2
31	21573	60839	34330	5445	EDA7	861A
32	13112	46370	59385	3338	B522	E7F9
33	13110	54479	39971	3336	D4CF	9C23
34	12866	44641	48857	3242	AE61	BED9
41	21573	60060	24692	5445	EA9C	6074
42	13111	21315	40654	3337	5343	9ECE
43	13110	44536	5265	3336	ADF8	1491
44	12609	9680	60313	3141	25D0	EB99
51	21573	49116	17537	5445	BFDC	4481
52	13112	30649	773	3338	77B9	305
53	13110	12339	60154	3336	3033	EAFA
54	13122	55736	36961	3342	D9B8	9061
61	21573	50212	3713	5445	C424	0E81
62	13112	15801	29590	3338	3DB9	7396
63	12848	16806	13890	3230	41A6	3642
64	13122	1280	1371	3342	500	055B
71	21573	20766	38525	5445	511E	967D
72	13110	42315	29840	3336	A54B	7490
73	13105	18337	37753	3331	47A1	9379
74	14147	42042	64641	3743	A43A	FC81
81	21573	43204	46221	5445	A8C4	B48D
82	13108	34745	29089	3334	87B9	71A1
83	12848	17297	64604	3230	4391	FC5C
84	13633	51485	49435	3541	C91D	C11B
91	21573	38238	17402	5445	955E	43FA
92	13107	28873	58404	3333	70C9	E424
93	12848	54804	14153	3230	D614	3749
94	13123	1034	45641	3343	040A	B249



Gambar 7.
Arsitektur CBC yang Dirancang
untuk Proses Dekripsi

Maka perbandingan *plaintext* dan *ciphertext* dari contoh ini ditunjukkan pada Tabel 8 berikut.

Tabel 8.
Perbandingan *Plaintext* dan *Ciphertext*
Contoh Awal

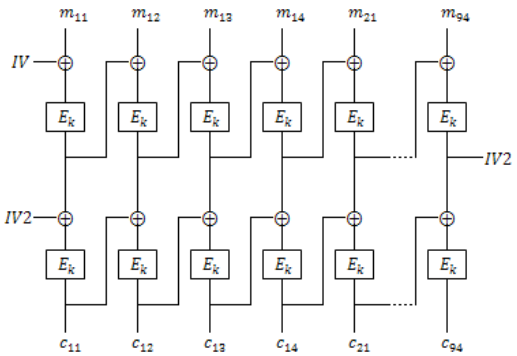
Plaintext	*Ciphertext*
23206019	BE84ABE9D0F5CAAC
TE38365A	2EC4FDEA1468B9E2
TE38362B	861AE7F99C23BED9
TE37361A	60749ECE1491EB99
TE38363B	44810305EAFA9061
TE38203B	0E8173963642055B
TE36317C	967D74909379FC81
TE34205A	B48D71A1FC5CC11B
TE33203C	43FAE4243749B249

ika pada *plaintext* nilai dari mata kuliah TE38363 diubah dari B menjadi C, maka hasilnya akan menjadi seperti pada Tabel 9.

Maka terlihat masih terdapat kekurangan dari proses yang dihasilkan. Diharapkan proses kriptografi ini memiliki sifat *diffusion* dimana setiap bit dari setiap *plaintext* mempengaruhi setiap bit *ciphertext*. Tetapi dari ilustrasi di atas ternyata perubahan satu bit dari *plaintext* pada baris ke-5 hanya mempengaruhi setiap bit *ciphertext* yang berada pada baris ke-6 hingga ke-9. Sedangkan bit *ciphertext* yang berada pada baris ke-1 hingga ke-5 tidak terpengaruhi oleh perubahan bit yang berada pada baris ke-5. Agar sifat *diffusion* ini dapat tercapai, maka dirancang arsitektur operasi CBC berulang menjadi seperti Gambar 8.

Tabel 9.
Perbandingan *Plaintext* dan *Ciphertext*
Setelah Perubahan 1 Bit *Plaintext*

Plaintext	*Ciphertext*
23206019	BE84ABE9D0F5CAAC
TE38365A	2EC4FDEA1468B9E2
TE38362B	861AE7F99C23BED9
TE37361A	60749ECE1491EB99
TE38363C	44810305EAFA9A7F
TE38203B	340BC16211739A50
TE36317C	30A689250000D395
TE34205A	24FFDD9AD40E37A8
TE33203C	6F5B8F3FE426E77D



Gambar 8.
Modifikasi Operasi CBC untuk
Menambahkan Sifat *Diffusion*

Pada arsitektur tersebut, terdapat dua kunci yaitu *IV* dan *IV2*. Nilai *IV2* akan sama dengan nilai c_{94} pada arsitektur operasi CBC seperti pada Gambar 6**.** Pada contoh kasus ini, dengan nilai e = 163, n = 66203, dan *IV* = 3139; maka diperoleh *IV2* = B249.

Dengan arsitektur CBC berulang ini, perbandingan *plaintext* dan *ciphertext* dari contoh ini ditunjukkan pada Tabel 10. Sedangkan jika pada *plaintext* nilai dari mata kuliah TE38363 diubah dari B menjadi C, seperti kasus sebelumnya, maka per-

bandingan *plaintext* dan *ciphertext* ditunjukkan pada Tabel 11.

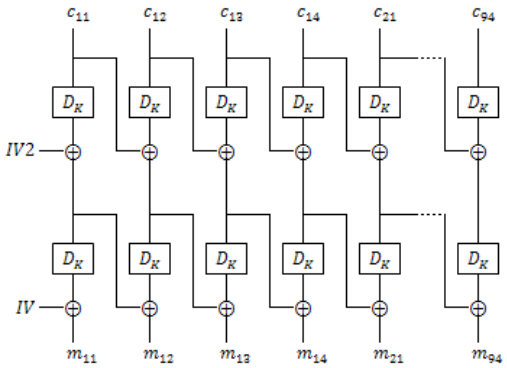
Tabel 10.
Perbandingan *Plaintext* dan *Ciphertext* dengan Arsitektur

Plaintext	*Ciphertext*
23206019	B8FB21BE384410E6
TE38365A	FDA721FD551FD29F
TE38362B	5B5792F561053C25
TE37361A	9CDE5FAEC2CFDF2A
TE38363B	F95FB42555D0335B
TE38203B	196CE1917EBA57DD
TE36317C	155BB5D629FF682F
TE34205A	3351B1B1B9F0D4F5
TE33203C	D6C9ECE6FA19021E

Tabel 11.
Perbandingan *Plaintext* dan *Ciphertext* setelah Perubahan 1 Bit *Plaintext*

Plaintext	*Ciphertext*
23206019	9877A60D29F21086
TE38365A	F12A56CC069DC9A2
TE38362B	13CD99E9B49B8D06
TE37361A	8ABF88118505A97D
TE38363C	A9F6612B43EF5B38
TE38203B	A552F8F9A5284C1D
TE36317C	06A72BD6B590BF6C
TE34205A	056B464FE4B4C0D8
TE33203C	CE5279C8364B043C

Terlihat bahwa sifat *diffusion* telah berhasil diterapkan melalui arsitektur CBC berulang ini. Untuk proses dekripsi, maka digunakan arsitektur seperti Gambar 9.



Gambar 9.
Proses Dekripsi Modifikasi Operasi CBC untuk Menambahkan Sifat *Diffusion*

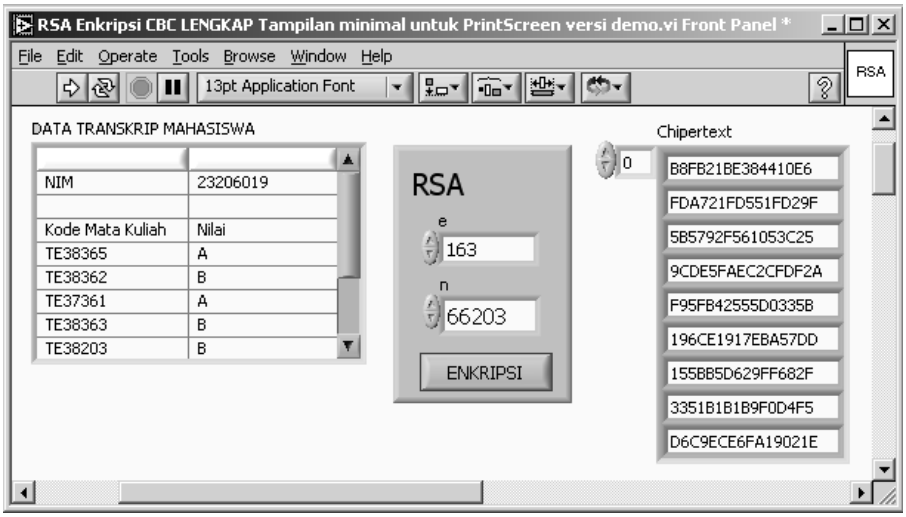
IMPLEMENTASI DAN ANALISA DATA

Untuk implementasi algoritma RSA dalam pengamanan data transkrip akademik mahasiswa ini, dibuat suatu perangkat lunak berbasis LabVIEW 7.1. Hasil contoh tampilan perangkat lunak yang dibuat adalah ditunjukkan pada Gambar 10 dan Gambar 11.

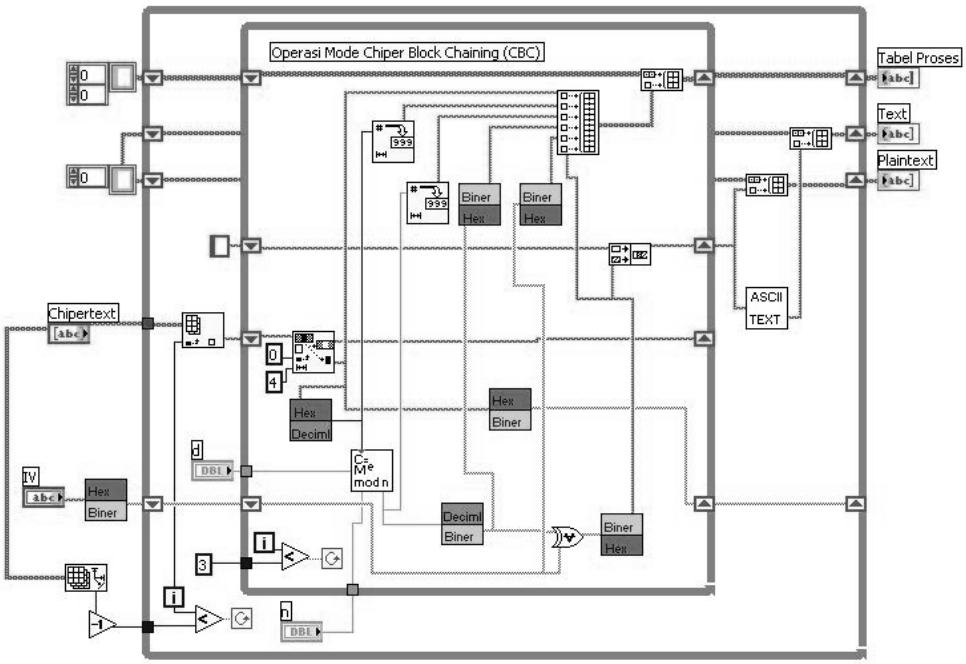
Pada penelitian ini, kinerja yang diukur adalah waktu komputasi dan kebutuhan memori yang digunakan dalam komputasi algoritma RSA. LabVIEW menyediakan fasilitas profil untuk mengamati kedua hal tersebut. Hasil profil dari program RSA yang dibuat ditunjukkan pada Gambar 12 dan Gambar 13. Terlihat bahwa waktu komputasi RSA untuk mengenkripsi data transkrip akademik mahasiswa adalah 15625 mikrodetik dengan kebutuhan memori adalah 3908 bytes.

KESIMPULAN

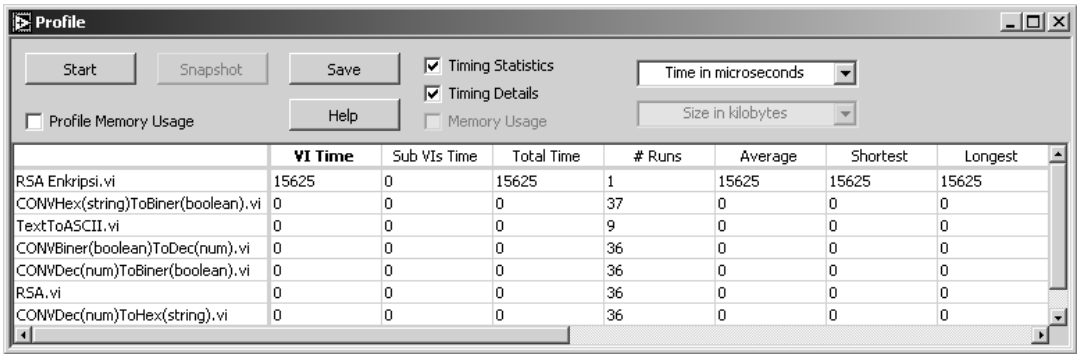
Berdasarkan hasil implementasi, dapat disimpulkan bahwa algoritma RSA telah dapat diaplikasikan dalam pengamanan data transkrip akademik mahasiswa. Penerapan operasi *Chiper Block Chaining* pada algoritma kriptografi RSA terutama ditujukan untuk menghasilkan prinsip diffusion, dimana setiap bit dari setiap *plaintext* mempengaruhi setiap bit *ciphertext*. Waktu kom-



Gambar 10. *Front* Panel Implementasi RSA pada Enkripsi Data Transkrip Mahasiswa

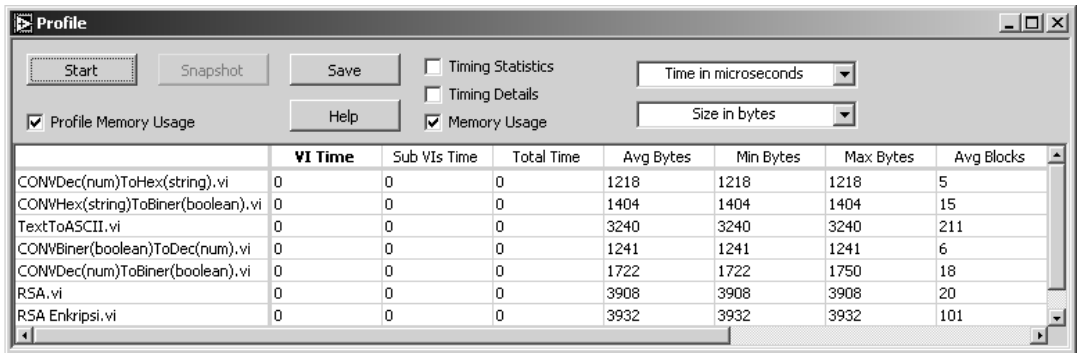


Gambar 11. Blok Diagram Implementasi RSA pada Enkripsi Data Transkrip Mahasiswa



	VI Time	Sub VIs Time	Total Time	# Runs	Average	Shortest	Longest
RSA Enkripsi.vi	15625	0	15625	1	15625	15625	15625
CONVHex(string)ToBiner(boolean).vi	0	0	0	37	0	0	0
TextToASCII.vi	0	0	0	9	0	0	0
CONVBiner(boolean)ToDec(num).vi	0	0	0	36	0	0	0
CONVDec(num)ToBiner(boolean).vi	0	0	0	36	0	0	0
RSA.vi	0	0	0	36	0	0	0
CONVDec(num)ToHex(string).vi	0	0	0	36	0	0	0

Gambar 12. Profil VI yang Menyajikan Waktu Komputasi RSA dalam Mikrodetik



	VI Time	Sub VIs Time	Total Time	Avg Bytes	Min Bytes	Max Bytes	Avg Blocks
CONVDec(num)ToHex(string).vi	0	0	0	1218	1218	1218	5
CONVHex(string)ToBiner(boolean).vi	0	0	0	1404	1404	1404	15
TextToASCII.vi	0	0	0	3240	3240	3240	211
CONVBiner(boolean)ToDec(num).vi	0	0	0	1241	1241	1241	6
CONVDec(num)ToBiner(boolean).vi	0	0	0	1722	1722	1750	18
RSA.vi	0	0	0	3908	3908	3908	20
RSA Enkripsi.vi	0	0	0	3932	3932	3932	101

Gambar 13.
Profil VI yang Menyajikan Kebutuhan Memori yang Digunakan dalam Komputasi RSA

putasi RSA adalah 15625 mikrodetik. Kebutuhan memori komputasi RSA adalah 3908 bytes. Tentu saja yang tidak kalah pentingnya adalah pengaturan manajemen kunci dan distribusi pasangan kunci sehingga protokol keamanan ini dapat berjalan dengan baik.

DAFTAR PUSTAKA

Hafni Syaeful Sulun, *Penerapan Algoritma Kriptografi RSA dalam Pengiriman Data Melalui Socket Berbasis TCP/IP*,

Ir. Rinaldi Munit, M.T., *Algoritma Pendukung Kriptografi*

Ir. Rinaldi Munir, M.T., *Algoritma RSA dan ElGamal*

Ir. Rinaldi Munir, M.T., *Tipe dan Mode Algoritma Simetri*

Inu Laksito Wibowo, *Aplikasi Algoritma RSA Pada Sistem Pengaman Data Yang Menjamin Keaslian dan Kerahasiaan Data*, KAPPA (2001) Vol. 2, No. 2, 12 - 20

Iswanti Suprapti, *Studi Sistem Keamanan Data Dengan Metode Public Key Cryptography*

Puja Pramudya, *Aplikasi Kriptografi Untuk Keamanan Pelaporan Pemungutan Suara Pada Pemilihan Umum Presiden Berbasiskan Layanan Pesan Pendek di Indonesia.*

Supriyono, *Pengujian Sistem Enkripsi-Dekripsi Dengan Metoda RSA Untuk Pengamanan Dokumen*, JFN, Vol 2, No. 2, November 2008